Algorithm
FindMaximumRegion

Purpose: From a region (*smin*, *smax*) with singular mapping to action *a* in *tree*, find a *smax'* such that the region (*smin*, *smax'*) has the following properties:

- singular mapping to action *a* in *tree*
- it does not cause any other region in *tree* to be split in more than two
- (it does not overlap with previously found regions)
- no increment of *smax'* in any dimension can retain the above properties

Requires:

tree -> a decision tree inducing the partitioning to minimize
bounds -> a nested list of bounds in each dimension, sorted in ascending order
pmin, *pmax* -> vectors of indices pointing to values in *bounds*
smin, *smax* -> points in the state space that together defines (the remains of) a region in the original partitioning. Corresponds to the values in *bounds* pointed to by *pmin* and *pmax* respectively
trackTree -> a decision tree used to track the areas of the state space already covered by previous iterations

Assumes functions:

MakeState(*p*, *bounds*) -> create a state *s*, where *s_i* = *bounds_{i, p_i}*
IsExplored(*smin*, *smax*, *tree*) -> return **true** if any part of the region (*smin*, *smax*) has decision value in *tree*
ActionsInRegion(*smin*, *smax*, *tree*) -> return the set of actions assigned to the region (*smin*, *smax*) in *tree*
GetBroken(*smin*, *smax*, *tree*) -> return a list of leaves in *tree* that the region (*smin*, *smax*) splits in more than 2
MarkAsExhausted(*dim*) -> mark dimension *dim* as exhausted

```
healing <- false
candPmax <- pmax
while unexhausted dimensions do

  do expansion in some dimension if we are not healing broken regions
  if not healing then
    dim <- choose unexhausted dimension
    prevPmaxInDim <- candPmax_dim
    candPmax_dim <- candPmax_dim + 1
  endif

  check the result of performing the expansion
  candSmax <- MakeState(candPmax, bounds)
  explored <- IsExplored(smin, candSmax, trackTree)
  actions <- ActionsInRegion(smin, candSmax, tree)
  broken <- GetBroken(smin, candSmax, tree)

  case 1: simply undo the expansion
  if explored or actions != {a} then
    healing <- false
    MarkAsExhausted(dim)
    candPmax_dim <- prevPmaxInDim

  case 2: start healing process and don't undo last expansion...
  else if broken is not empty then
    healing <- true
    prevPmaxInDim <- candPmax_dim
    candPmax_dim <- maximum bound on dimension dim for any region in broken

  ...or accept that we can't heal the regions and reset to last accepted pmax
    if prevPmaxInDim = candPmax_dim then
      healing <- false
      MarkAsExhausted(dim)
      candPmax <- pmax
    endif

  case 3: the expansion was successful, so update the accepted pmax (and exhaust dim if we are on the edge)
  else then
    healing <- false
    pmax <- candPmax

    if pmax_dim is the maximum bound on dimension dim then
      MarkAsExhausted(dim)
    endif

  endif
endwhile

return pmax
```